Click here to view PowerPoint presentation; Press Esc to exit

Combat Modeling by Using Simulation Components

"Today it is not the word "truth" but the word "model" that continually decorates the pages of scientific journals."
Owen Gingerich (1982)

Mr. Oray Kulaç / Mr. Murat Gúnal
Turkish Naval HQs R&D and Decision Support Department
Deniz Kuv.Kom. APGE, Bakanliklar
06100 Ankara-TURKEY
Email: kulac@dzkk.tsk.mil.tr / gunal@dzkk.tsk.mil.tr

ABSTRACT

Growing cost of operational trials and limited budgets make extensive live tests of military systems and frequent live exercises nearly impossible. Also defense planning dealing with extremely complex military system behaviors necessitates the utilization of innovative analysis tools. Simulation is one of the most employed tools for the military analyses, training and acquisition. Its increasing importance is now well-recognized in most of the armed forces and in NATO. Dramatic advances are being made in military simulation community, but these advances are associated mainly with computer technologies.

Modeling and simulation efforts are subject to some problems. First of all, building and using a simulation model is an expensive and laborious activity. It is difficult to design a model to efficiently and effectively support the appropriate analyses. Also, models are generally built in closed architectures and are difficult to adapt to the specifics of different, however related, analysis efforts. Turkish Navy has started a research project in order to minimize above deficiencies. This research effort concerns at first with the simulation model infrastructure and then the model itself. The research on simulation model infrastructure exploits the theory and methodology that will make it possible to design and construct reusable and flexible models.

The results of this study are going to be the development of a flexible, scalable and reusable tool that can be used for the construction of a library of Naval combat model components. The JAVA™ programming language has been chosen for implementation in order to take advantage of world-wide web. The purpose of Operations Research (OR) type of combat components is to provide a library of reusable software to speed development of OR applications and make them more reliable. The component architecture, which is in development phase, supports reuse, easy model configuration, interoperability, flexibility and scale changes in successive stages of analysis.

1. INTRODUCTION

The tasks that naval forces are required to perform have changed little over the decades. "However, existing instabilities in the international arena are being accentuated and new ones are appearing especially after the cold war era. Many conflicts arising from disputes over resources, ethnic and religious hatreds and drives for regional dominance can be expected. Thus, the future national security environment in which the naval forces will play a key part is likely to change dramatically "[Ref.1]. In addition to this fuzzy fast changing environment, technology is improving so rapidly as affecting weapon systems and spreading worldwide. Thus, a deterrent naval force will have to be alert for significant technological change and be ready to exploit new technologies.

What will be the impact of a new technology weapon system on naval vessels? This is one of the most central questions that a naval defense analyst must be asking to himself or herself. The growing cost of operational trials makes extensive live tests of new technology military systems nearly impossible. Analyses

Paper presented at the RTO NMSG Conference on "NATO-PfP/Industry/National Modelling and Simulation Partnerships", held in Paris, France, 24-25 October 2002, and published in RTO-MP-094.

Report Documentation Page				*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **00 NOV 2003**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Combat Modeling by Using Simulation Components**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Turkish Naval HQs R&D and Decision Support Department Deniz Kuv.Kom. APGE, Bakanliklar 06100 Ankara-TURKEY**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release, distribution unlimited**		
13. SUPPLEMENTARY NOTES **See also ADM001655., The original document contains color images.**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**20**	

of such extremely complex system behaviors necessitate the utilization of innovative tools that are both flexible and reusable.

Simulation is one of the most employed tools for the military analyses. Ideally military simulation modeling should be quick, cheap, and yield precise answers. "However, many current models are large, monolithic and hard to understand" [Ref.2]. Models are generally built in closed architectures and are difficult to adapt to different scenarios. They often tend to be large. Addition to these, building and using a simulation model is an expensive, slow and cumbersome activity. However, military analysts concerned with the efficiency of simulation modeling, meaning that combination of correctness, flexibility and affordability that supports military decision-making.

For this main reason combat simulation modeling needs different ways of modeling approaches and techniques. Mainly these approaches or techniques should provide the strength of

- Modularity
- Scalability
- Reusability
- Network awareness
- Platform independent behavior,

to a Combat Simulation model.

Component Based simulation modeling is a good modeling solution for Combat Simulation efforts. It provides most of the properties mentioned above. Following sections explains the architecture of component-based modeling.

2. COMPONENT BASED MODELING

Component Based Modeling is in some sense an evolution of object-oriented thinking. However, it differs from object-oriented modeling in several ways and does not require an object-oriented programming language for implementation. In the following section, object-oriented and component based modeling are discussed in the simulation modeling point of view.

2.1. Objects vs. Components

First of all in Object Oriented Modeling it is difficult to couple objects loosely. However, simulation components are specifically designed to provide this property. In Object Oriented design, inheritance and overloading are the primary mechanisms for implementing polymorphism. On the other hand, in Component Based design common interfaces between components are established.

Object-oriented development is both a top down and a bottom up process. However, component-based design is a pure bottom up process. Component based design is fast comparing to object-oriented design, but requires the existence of an already build library of proven components.

In component-based design one important aspect is the degree of loose coupling between components. This property enables components to act without considering the outer world and this extends the flexibility and modularity.

Listener pattern is a software design pattern that enables the loosely coupling. It is the primary mechanism for components to interact. In this pattern components signal their state changes by broadcasting events to outside world and the objects that are interested about these changes receive these events. As explained in Ref.3 two types of components are involved with a listener pattern: the Listener component and the Event Source component. The "listening" component registers interest in another component's events and waits for the other component to fire the event. When the event fires in the simulation component (that is, an event source component), it notifies all its registered listeners of the event. These events have no duration. The same component can serve as a Listener to some components and be an Event Source to other components. The Event that is fired should contain enough information for the Listener to be able to act without a callback to the Event Source. This no-callback property is a critical one for maximizing the looseness of the coupling between components. [Ref.3]

Component based design has been used a lot by computer hardware engineering and electrical engineering. A good example of a component system is the personal computer (PC). A PC has many parts (components) that communicate with each other. It is very easy to connect/disconnect or change components depending on need and use. Most importantly, all these components work together to form a complex system. Another example could be the electronic kits used in circuits. Simulation and Modeling community should try to benefit from components in the area of modeling as much as the other communities do.

A component is the basic element of component-based design. Although a satisfactory definition of component remains elusive, following section tries to explore a definition to simulation component.

2.2. What is a simulation component?

In his book "Beyond Objects: Components" Clemens Szyperski talks about components as follows "One thing can be stated with certainty: components are for composition. Nomen est omen. Composition enables prefabricated "things" to be reused by rearranging them in every new composites. Beyond that trivial observation, much is unclear." This sentence gives a sense of components even with no definition. [Ref.4]

According to OMG (Object Modeling Group) Modeling Language Specification "A component is a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces."

Bertrand Meyer in his on-line article "What to Compose (Jan 2000)" gives the seven criteria for composites. According to these criteria, components;

- Maybe used by other elements (clients)
- Maybe used by clients without the intervention of components developers
- Includes a specification of all dependencies (hardware and software platform, versions, other components)
- Includes a precise specification of the functionality it offers.
- Is usable on the sole basis of that specification
- Is composable with other components
- Can be integrated into a system quickly and smoothly

Pidd [Ref.5] also makes a brief definition of a software component for discrete simulation and indicates the following criteria: Component's functionality should be entirely defined, all communication with any other components should be through a fully defined interface that is wholly unambiguous.

Arnold Buss in his paper "Component Based Simulation Modeling" defines component as a monolithic programming entity whose external interface consists only of property accessor/mutator methods, of action methods, and event handler methods [Ref.3]. This definition seems more specific than the others and focuses more on implementation. In this definition, *Property accessor/mutator* methods are small methods whose only purpose is to enable reading/writing a single property. Commonly used synonyms are "setters" and "getters" for these methods. An *event handler method* is a method that supports the Listener Pattern discussed above. Its signature is always the event of interest. An *action* method changes the state of the component in ways that are typically more complicated than simply setting the value of a property.

After all these definitions, a more general definition attempt could be as follows "A software component is an executable monolithic object designed to be easily replaceable as a unit in the context of system".

As seen in above definition attempt components are expected to be monolithic. However the term monolithic has a negative impact since it has been used to define legacy systems, which are too inflexible. A monolithic system is resistant to easy modification and is difficult to extend. Therefore, it is highly desirable for components since monolithic behavior prevents extension and represents solid structure and robustness.

Following properties are appeared to be the important properties of components. A software component;

- Should be able to work stand alone as separate entity,
- Should communicate with other components by passing messages,
- Should be able to provide data to each other,
- Must be composable,
- Must be loosely coupled,
- Should be simple,
- Must be of high quality,
- Should have a good documentation.

3. TURKISH NAVY AIR DEFENSE MODEL: AN EXAMPLE FOR COMPONENT-BASED SIMULATION

3.1. Architecture

As an operations analyst, one knows that effective analysis requires the simulation models to have flexibility, modularity, scalability and reusability. Another important feature is that the model should be independent of platform. In order to have such a model for Turkish Navy Air Defense analysis purposes component based simulation modeling approach has been chosen. JAVA programming language has been chosen for the implementation. A fundamental reason for selecting JAVA is the web technology. Web technology has the potential to significantly alter the ways in which simulation models are developed, documented, analyzed and executed. And Java programming language and its applications have substantially extended many capabilities for network based simulations. Pidd and Cassel [Ref.6, 7] also suggest the use of Java language for networked and web based simulation development.

Component architecture is developed by using the core structure, which is introduced in [Ref. 8]. As seen in figure 1, the combat component has a standardized way of sending or receiving messages from other combat components and processing these messages. This send/receive process is conducted by four connectors (pins). Two of these pins deal with incoming/outgoing events and the other two deals with incoming/outgoing properties. The following is the list of these pins and their duties,

Property user:	The *property user* is the part of the combat component that deals with incoming properties.
Property source:	*The property source* is the part of the combat component that deals with outgoing properties.
Event listener:	*The event listener* is the part of the combat component that deals with incoming events.
Event source:	*The event source* is the part of the combat component that deals with outgoing events.

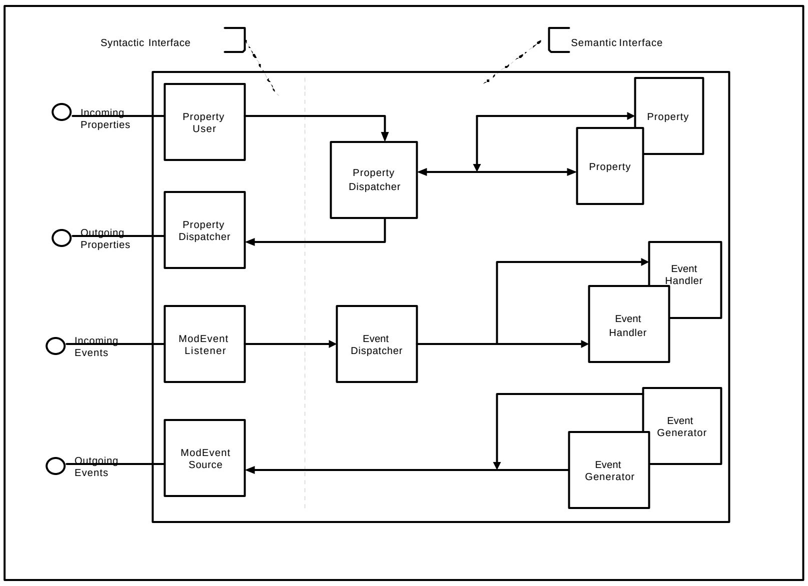


Figure 1. Basic Structure of Combat Component [Ref.8]

Definitions of the *event* and *property* will make the structure of the combat *component* more clearly to the reader. If a combat *component* wants to inform other components about a new change in its status, it will generate a message to all listener components without caring how the listeners react to that message. The content of this message is defined in the *event object* transmitted. This object is called an *event.* A *component* can generate and listen to *events*. A combat *component* is limited (monolith) in what it can do. Sometimes another component may tell a combat *component* what action to take or it may obtain some information from it. This action is implemented by using properties. A *property* is a piece of data that a component has, uses or can provide.

The combat components give us opportunity of building a library of combat components for fast and reliable modeling.

Another important structure of the component-based modeling is combat containers (Figure 2). A combat container is also a combat component that contains some other components and containers inside. Actual examples for Combat Containers are a plane, a ship, a guided missile, a tank etc. A combat container is the parent of all components it contains. This parent-child relation becomes important when composing complex systems. A modeler should only put the components that are created earlier into a container and do not think about their interaction inside the container. Two important properties of a container are its type and its side. Side is defined by a color code according to NATO Military Standards.

Discrete event simulation has been used [Ref.9, 10] for the scheduling purposes since it provides a flexible and descriptive modeling methodology, which is very convenient for our modeling purposes.

For geographical positioning purposes and visualization, Java based Geographical Information System (GIS) has been used. In order to maintain the flexibility the interaction between combat components and GIS has been designed in Model-View-Controller concept. In our case the Model is the Combat-Container library and it is the main body of the simulation. Since the model is made up of components and containers explaining these building blocks will reveal the simulation itself.

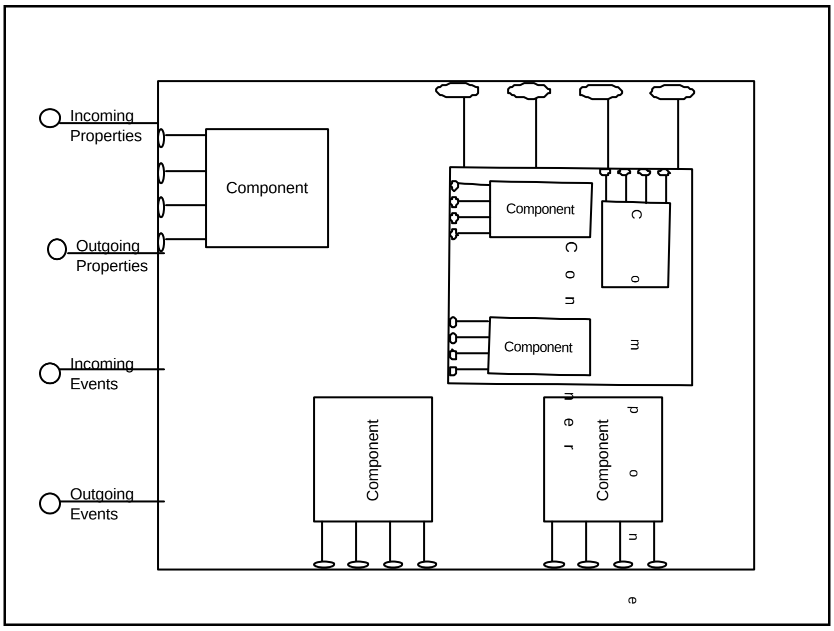


Figure 2. Structure of Combat Containers

The View is the GIS tool and it is composed of layers. For instance Digital Elevation Data (DTED) is a layer on the map. The Model interacts with View by way of a discrete event simulation layer and therefore it can reach GIS data easily. One of the reasons of this interaction is that a component, like active sensor, may need to calculate its Line Of Sight (LOS) by using one of the LOS algorithms existing in the literature all of which need terrain data.

The Controller is the scenario definition file for the simulation. Initial positions and parameter values of combat components are the some of the contents of the controller.

Following section provides structural information on Components and Containers of Turkish Navy Air Defense Model.

3.2. Components and Containers

This section mainly focuses on implementation rather than architecture. Three types of containers are created. These are Defender, Attacker and Defended Target. As discussed in previous sections a container is actually a combat component which can contain some other components. The component structure of the model is shown in figure3.

As seen in figure, all of the containers contain a common component named the mover component. Main functionality of the mover component is to keep track of the container's position according to its moving scheme. In our application the scheme is linear. It means that when a mover starts moving it proceeds with constant velocity and steady direction. Even though this is not realistic most of the time, this level of detail is accepted for our purposes. Future work consists of making the movement more realistic namely nonlinear. Perhaps in real life, a combat object usually moves nonlinearly which it can roll, yaw, pitch and accelerate.

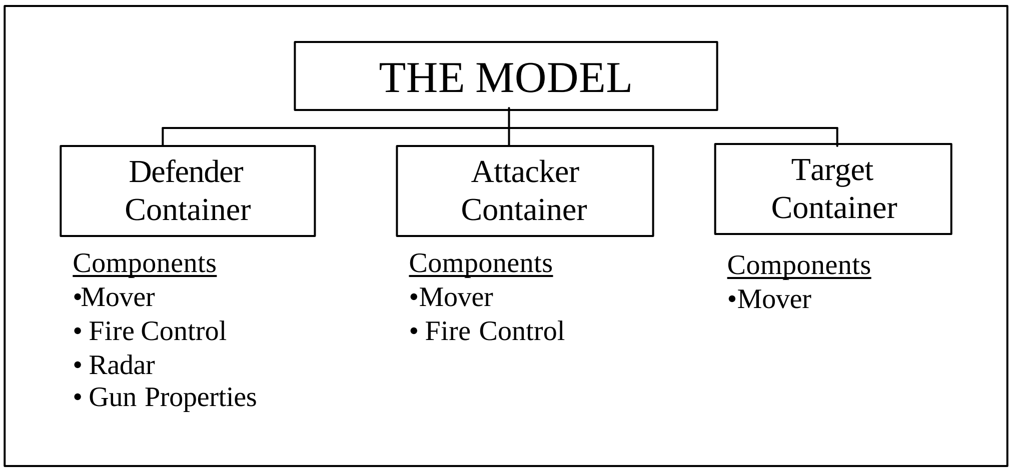


Figure 3. The Component Structure of the Model

The mover component is a property source and an event source. It has the necessary properties to keep location and velocity information. This will be the same for fixed components. In this case the mover component functions as a location marker. The mover component also fires necessary events in order to inform (notify) other components about position and state of the velocity changes. As an example, when a mover reaches to its given destination, it generates "At Location" event and notify listeners. Any event listener of the mover receives this message and reacts accordingly.

One of the most significant components in terms of functionality is the active sensor component. It is the implementation of radar in the model and a part of the defender container. Main function of it is to detect attackers in the surrounding area, such as cruise missiles or airplanes. Detection event is scheduled depending on line of sight of sensor, target's distance and detection probability of the sensor component.

In order to collect sample data from the simulation runs a listener component called " Stats Helper" is used. This component listens necessary events and keeps the necessary data about these events like time of the event.

Interaction between containers is handled by using small referee component, which evaluates the interaction between containers and gives a decision concerning this interaction. For example, interaction between radar and a target first investigated by a referee component which is created between these components and referee decides when the radar can see the target according to radar and target properties.

4. CONCLUSION

In this paper, we first presented a discussion on the proceedings of component based simulation and then made a formal definition for a simulation component. This is necessary to clarify our understanding on the terminology because of the difference between an object and a component may not be understood properly. Besides its theoretical side, we also presented an implementation that component based simulation concepts have been accepted. Remarkable point in our study is, as seen from the structure of the model, it is very easy to take out one of the components and put a different one in the model. This behavior provides analyst the necessary flexibility and speed. Thus, he/she can devote more time on analysis instead of trying to adjust an old model or building a new one.

REFERENCES

1. Committee on Technology for Future Naval Forces Naval Studies Board, "*Technology for the US Navy and Marine Corps Volume 1 Overview*", National Academy Press, U.S., 1997.

2. Committee on Technology for Future Naval Forces Naval Studies Board, "*Technology for the US Navy and Marine Corps Volume 2 Modeling and Simulation*", National Academy Press, U.S., 1997.

3. Buss, A.H., "Component Simulation Modeling Using Simkit," INFORMS National Meeting, Cincinnati, OH, May 2-5, 1999.

4. Szyperski C., "Component Software: Beyond Object Oriented Programming", Addison-Wesley Pub.Co., 1997, ISBN:020 1178 885.

5. Pidd M., Oses N., Brooks R., "Component-Based Simulation On The Web?"

6. Pidd M., Cassel R., 2000, "Web based simulation using Java", IEEE Potentials, vol 19 (1), pp 11-16.

7. Pidd M, Cassel R, 2000, "Using Java to develop discrete event simulations", Journal of the Operational Research Society, vol 51 (4), pp 405-412, ISSN: 01605682

8. Arent A., U.S.Naval Post Graduate School Thesis Study, Naval Post Graduate School, CA U.S., 1998.

9. Stork K., U.S.Naval Post Graduate School Thesis Study, Naval Post Graduate School, CA U.S.1996.

10. Buss A, Stork A.K., "Discrete Event Simulation on The World Wide Web using Java", Proceedings of the 1996 Winter Simulation Conference, pp 780-785., 1996.

11. Buss, A.H., "Component-Based Simulation Modeling," Proceedings of the 2000 Winter Simulation Conference, eds. Joines, J.A., Barton, R.R., Kang, K., and Fishwick, P.A., Piscataway, NJ, 2000, pp. 964-971.

Combat Modelling By Using Simulation Components

Authors: Lt. Oray Kulac / LtJG. Murat Gunal
Turkish Naval HQs R&D and Decision Support Department
Ankara-TURKEY
Email: kulac@dzkk.tsk.mil.tr / gunal@dzkk.tsk.mil.tr

PRESENTER: LtJG. MURAT GUNAL

UNCLASSIFIED

INTRODUCTION

Why Simulation?

Combat Modeling Problems

- Modularity, scalability and reusability

Next Sections

- Component Based Modeling Concept
- An Example To This Concept

COMPONENT - Definition

Some real examples of component based thinking

Definitions of Component

Clemens Szyperski: *"One thing can be stated with certainty: components are for composition. Composition enables prefabricated "things" to be reused by rearranging them in every new composites. Beyond that trivial observation, much is unclear."*

COMPONENT - Definition

Arnold Buss: Component is a monolithic programming entity whose external interface consists only of property accessor/mutator methods, of action methods, and event handler methods.

Mike Pidd: Makes a brief definition of a software component for discrete simulation and indicates the following criteria: Component's functionality should be entirely defined, all communication with any other components should be through a fully defined interface that is wholly unambiguous.

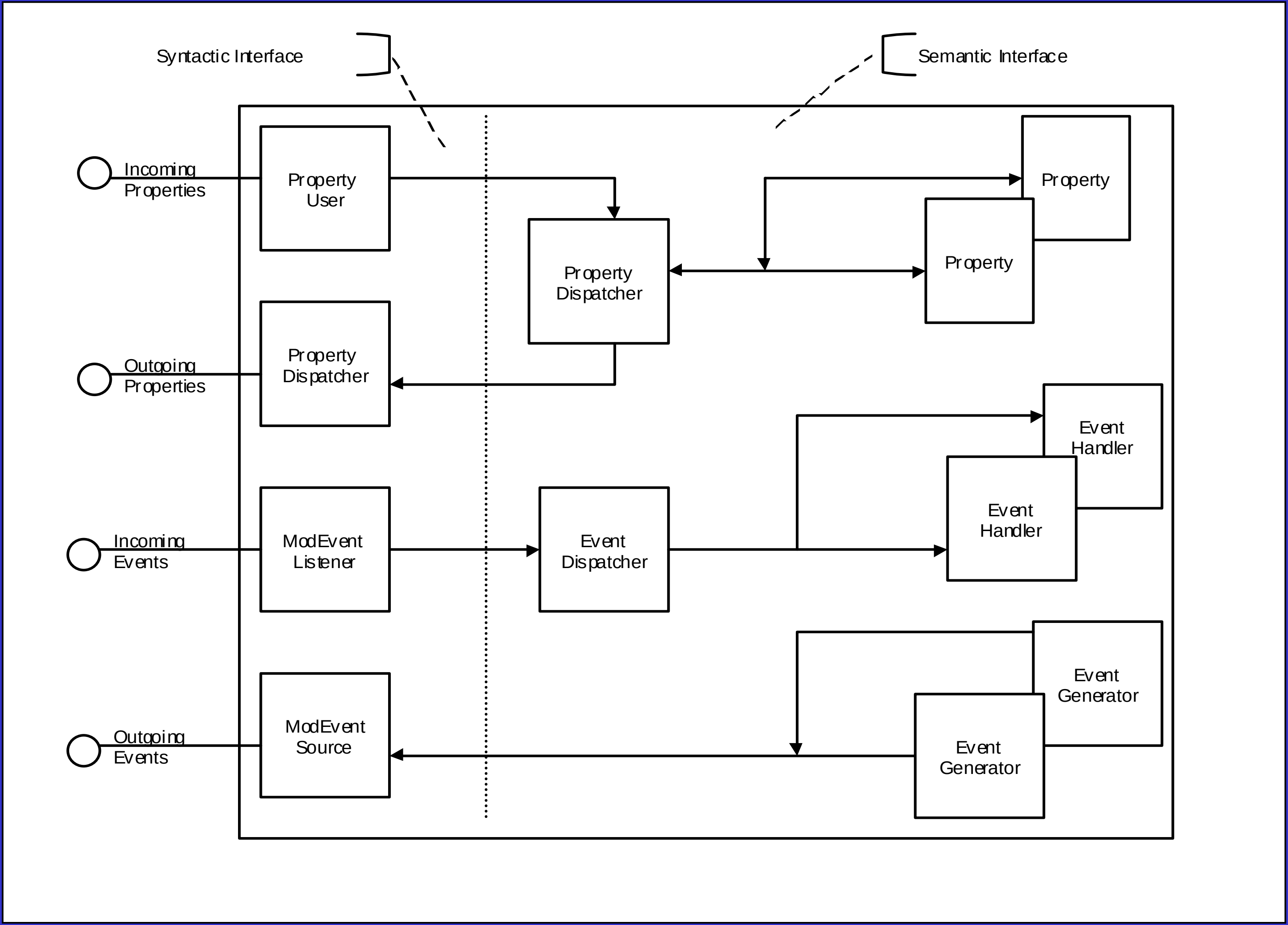
COMPONENT - Definition

A software component;

- Should be able to work stand alone as separate entity,
- Should communicate with other components by passing messages,
- Should be able to provide data to each other,
- Must be composable,
- Must be loosely coupled,
- Should be simple,
- Must be of high quality,
- Should have a good documentation.

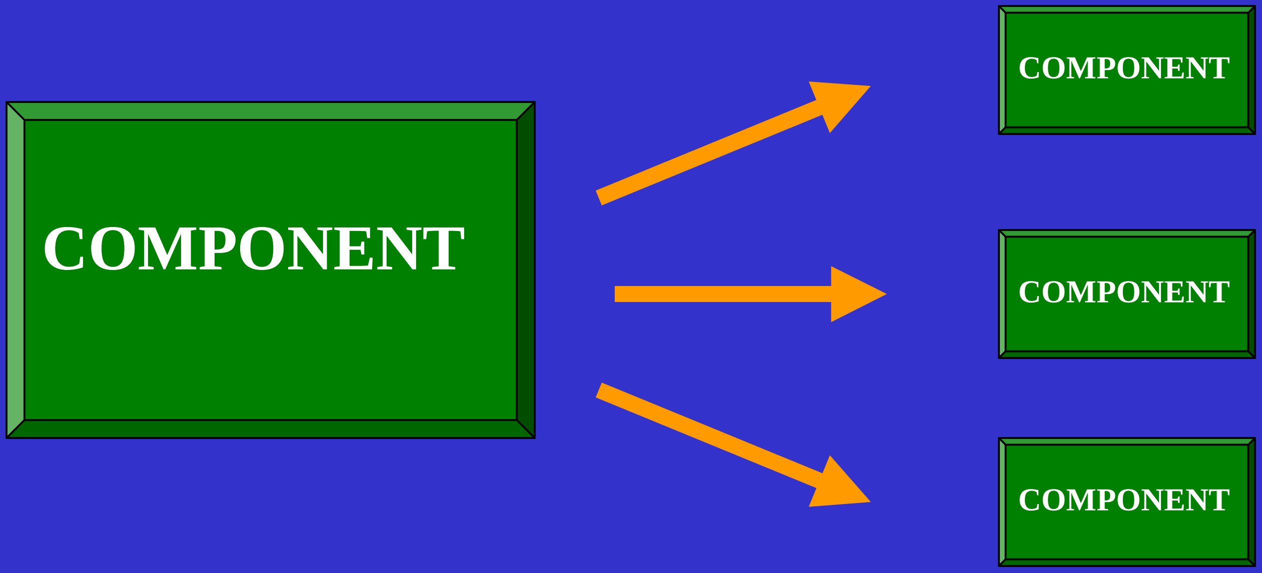
Object-Oriented vs Component Based Thinking

COMPONENT - Architecture



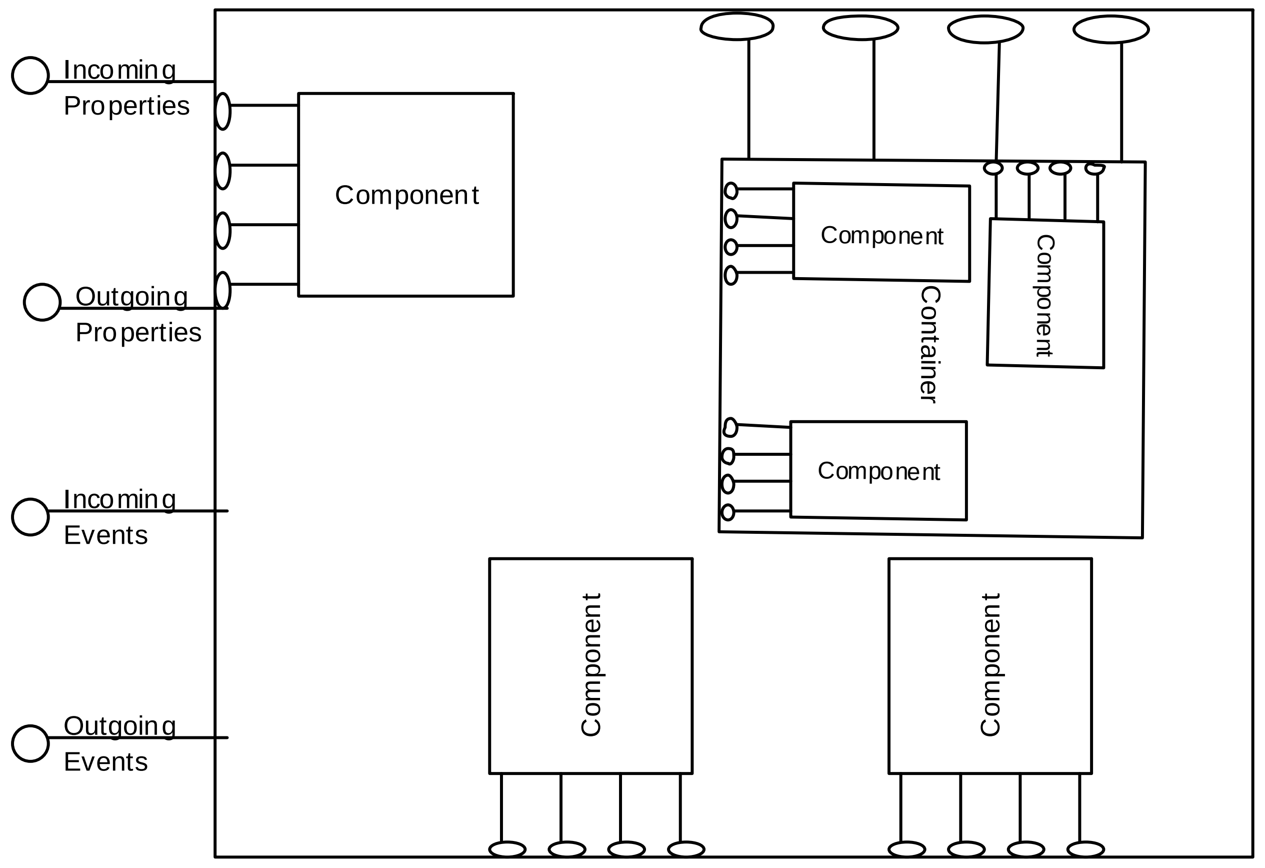
COMPONENT - Architecture

Notify Events
To all listeners



Notify Properties
To all users

CONTAINER - Architecture

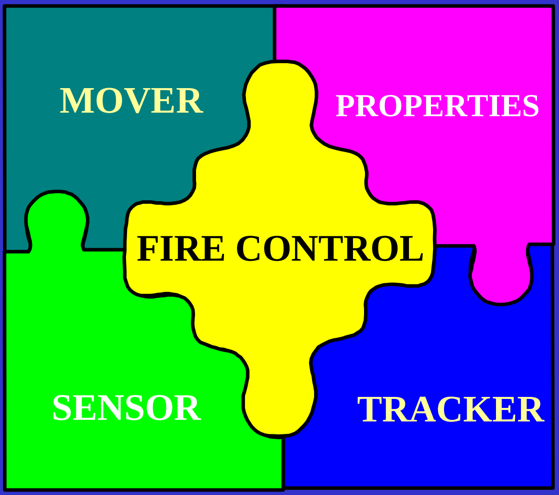


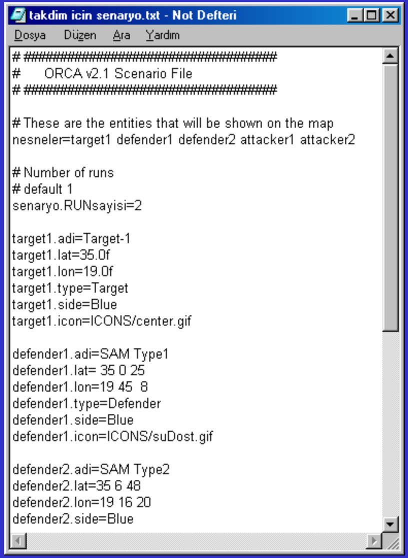
TURKISH NAVY AIR DEFENCE MODEL

JAVA Language

GIS Tool

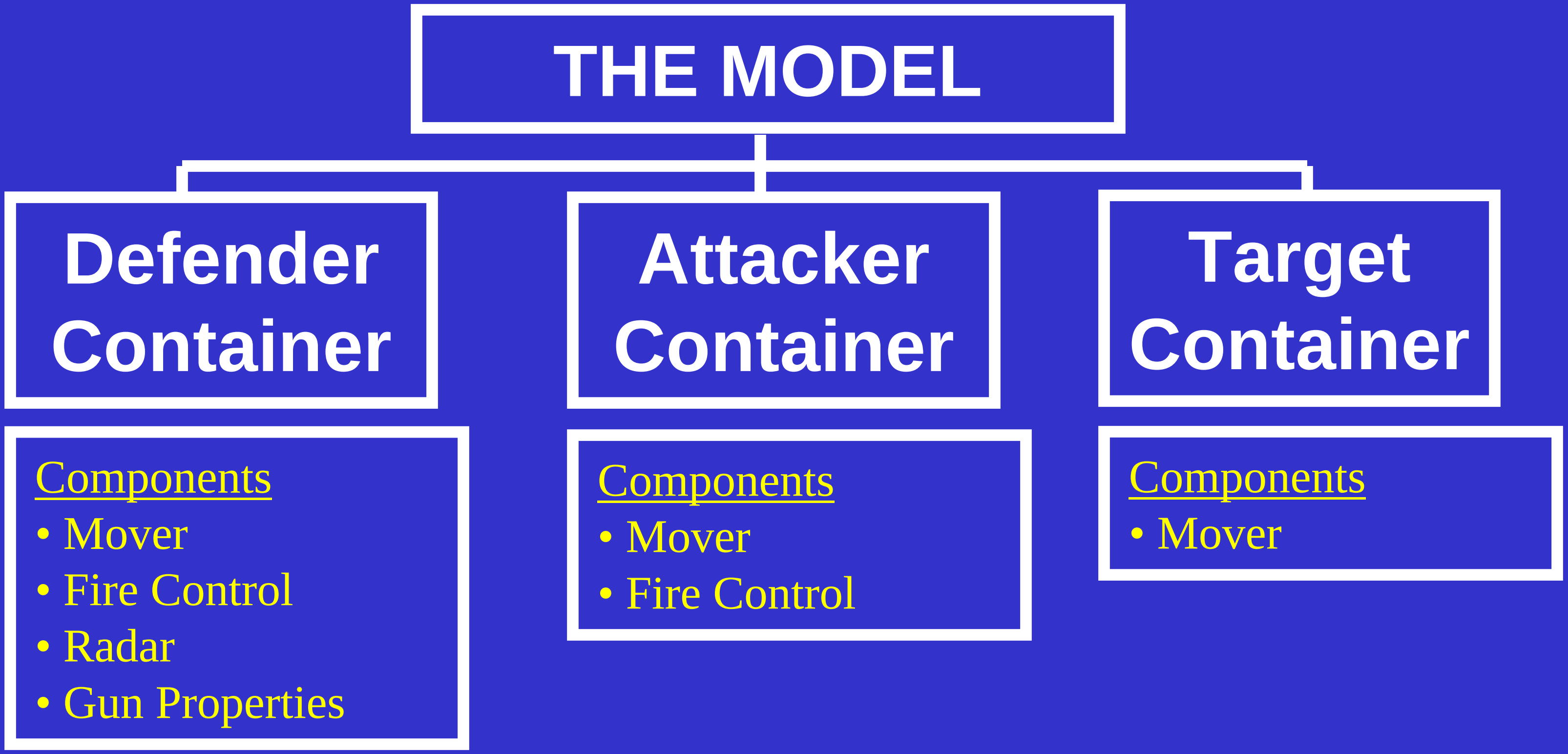
Model - View - Controller





```
# ########################################
#      ORCA v2.1 Scenario File
# ########################################

# These are the entities that will be shown on the map
nesneler=target1 defender1 defender2 attacker1 attacker2

# Number of runs
# default 1
senaryo.RUNsayisi=2

target1.adi=Target-1
target1.lat=35.0f
target1.lon=19.0f
target1.type=Target
target1.side=Blue
target1.icon=ICONS/center.gif

defender1.adi=SAM Type1
defender1.lat= 35 0 25
defender1.lon=19 45  8
defender1.type=Defender
defender1.side=Blue
defender1.icon=ICONS/suDost.gif

defender2.adi=SAM Type2
defender2.lat=35 6 48
defender2.lon=19 16 20
defender2.side=Blue
```

TURKISH NAVY AIR DEFENCE MODEL



Common Component: **The Mover Component**

-Currently Linear, should be nonlinear

TURKISH NAVY AIR DEFENCE MODEL

Mover Component

- Property Source: “Max Speed, Current Location”
- Event Source: “At Location, Stop Moving”

Sensor Component

- Line Of Sight (LOS) Calculation
- Detection Probabilities

Stats Helper

- Event Loging

Referee Component

CONCLUSION

Flexibility and Modularity in Model Building

Faster Design Time

But A Lot More To Do ...

THANKS